Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 11, No. 3, pp. 419–447, July 2015

THE MESH ADAPTIVE DIRECT SEARCH ALGORITHM WITH TREED GAUSSIAN PROCESS SURROGATES

Robert B. Gramacy and Sébastien Le Digabel*

Abstract: This work introduces the use of the treed Gaussian processes (TGP) as a surrogate model within the mesh adaptive direct search (MADS) framework for constrained blackbox optimization. It extends the surrogate management framework (SMF) to nonsmooth optimization under general constraints. MADS uses TGP in two ways: one, as a surrogate for blackbox evaluations; and two, to evaluate statistical criteria such as the expected improvement and the average reduction in variance. The efficiency of the method is tested on five problems: a synthetic one with many local optima, a synthetic one implying a numerical method, one real application from a chemical engineering simulator for styrene production, one from contaminant cleanup and hydrology, and one multidisciplinary design optimization application. In all five cases we show that the TGP surrogate is preferable to a quadratic model and to MADS without any surrogate at all.

Key words: *blackbox optimization, treed Gaussian processes (TGP), mesh adaptive direct search (MADS), surrogate management framework (SMF)*

Mathematics Subject Classification: *90C30, 90C56, 65K05, 60G15*

1 Introduction

We consider the following problem

$$\min_{x\in\Omega} \quad f(x), \tag{1.1}$$

where $\mathcal{X}$ is a subset of $\mathbb{R}^n$ ($\mathcal{X} \subset \mathbb{R}^n$), and $\Omega = \{x \in \mathcal{X} : c_j(x) \le 0, j \in J\} \subset \mathbb{R}^n$ denotes the feasible region. The functions f and c_j are defined by $f, c_j : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ for all $j \in J = \{1, 2, \ldots, m\}$. The two sets $\mathcal{X}$ and Ω are used in order to distinguish between two types of constraints: The set $\mathcal{X}$ may contain non-quantifiable and/or unrelaxable constraints, i.e., any constraint for which a measure of the violation is not available, and/or constraints that cannot be relaxed. Typically $\mathcal{X}$ contains bound constraints. On the other hand, the set Ω is defined by general constraints such as $c_j(x) \le 0$, $j \in J$, which are quantifiable and relaxable, meaning that they admit a distance to feasibility and that infeasible points may be acceptable as intermediate iterates on a path in search of a final optimal, and feasible, solution. Any solution $x \in \mathbb{R}^n$ has first to belong to $\mathcal{X}$ so that $f(x)$ and $c_j(x)$, $j \in J$, can be evaluated.

We focus on *blackbox optimization*, which occurs when both the objective function and constraints are available for evaluation only, not for inspection. That is, they are blackboxes, opaque to the optimization routine. A typical example of a blackbox is a complex computer

*The second author was supported by NSERC grant 418250 and AFOSR FA9550-12-1-0198.

simulation. They may exhibit characteristics that present unique optimization challenges, such as costly evaluations, noise, many local optima, and hidden constraints, which occur when the blackbox fails to evaluate, even at an *a priori* feasible point in Ω. As a consequence, and most importantly, derivative information is not present and impossible, and possibly even unhelpful, to approximate. This precludes the use of derivative-based optimization methods. Therefore derivative-free optimization (DFO) algorithms must be considered. A complete review of DFO methods is the subject of the recent book [21], in which DFO methods are classified as (quadratic) model-based methods and directional direct search methods, including the mesh adaptive direct search (MADS) algorithm [6] considered in the present study. The operations research community also considers mixing direct search methods and heuristics such as in [4,36,39,61]. Stochastic Nelder-Mead has also been studied in [15].

One advantage of directional direct search methods is that they allow the integration of secondary information and/or strategies in order to help diversify the search. One example involves the use of quadratic models to help accelerate the search [20, 23]. A disadvantage of this approach stems from the fact that quadratic models are local, which relegates their usefulness to smoother functions. In a way, the incorporation of quadratic models may be tantamount to estimating derivative information (at least to a second order), and as a result the final hybrid procedure may be more aggressive, i.e., less diversified, than a DFO-only method. A strategy more global and more robust to nonsmoothness and noise is desired in the context of blackbox optimization.

The surrogate management framework (SMF) [12] introduces the use of *surrogates* within directional direct search methods. Surrogates are global models such as Gaussian processes (GPs) [52], better known in the optimization community as kriging [41]. The idea is to fit a statistical model, and use its *mean* predictive surface as a surrogate for the actual evaluation of the blackbox. Traditional optimization methods can be used to find solutions to the surrogate optimization problem, in place of actual, expensive, blackbox evaluation(s). These optima are in turn taken as good candidates for the true functions. In order to simplify the presentation we use the term *surrogate* for the more longwinded "predictive mean surface of a fitted model". See Chapter 12 of [21] for a recent review of surrogates within DFO. One of many examples of a recent application using the SMF for a real-life application can be consulted in [45]. Other examples in [13, 53] respectively include the use of local linear models, for the constrained case, using a filter-like approach, and radial basis functions are considered for global optimization for bound-constrained problems with several local optima.

A criticism of these approaches echoes that for quadratic models. Although more sophisticated than quadratics, without the incorporation of the full predictive surface uncertainty, the search of the surrogate surface is bound to take on a local and greedy flavor. A fully statistical, and global, approach to optimization would leverage a probabilistic quantification of uncertainty about the potential for newly polled points to provide an improvement over the current best guess of the location of optima. We posit that using a full characterization of a fitted model's predictive surface to inform the search, not just the mean, represents a more conservative and therefore reliable approach. Relevant optimization statistics derived from the full predictive distribution include the *expected improvement* (EI) [40], on which we shall say more in due course.

One of the aims of this paper is to explore how such statistics can be used within a hybrid optimization scheme involving direct searches, surrogates model fits, and statistical

The statistics community also uses the term *kriging*, but to refer (again) to the predictive equations from the fitted model, not to the underlying probabilistic (Gaussian) process—a subtle distinction.

search criteria derived from the predictive distribution (in lieu of, or in addition to, surrogate optimization subroutines). In particular, we illustrate how aspects of the predictive variance (through EI or on its own), which is known to be well-estimated by GP surrogates (e.g., see [37]), can be at least as useful as the traditionally utilized predictive mean. Another is to propose the use treed Gaussian process (TGP) surrogates [28], which offer attractive benefits over the traditional GP ones, on which we shall elaborate further shortly. TGP has been used fruitfully in conjunction with the asynchronous parallel pattern search (APPS) [34] direct search method in an unconstrained optimization setting [57], and with generalized pattern search (GPS) [59] for a particular application [16]. This shows promise for the present paper, where (as a final layer in the contribution) we promote using MADS instead of APPS or GPS, for three main reasons: First, MADS offers a more diversified set of search directions resulting in a more efficient method as already demonstrated for example in [3]. Second, MADS deals natively with general constraints using the *progressive barrier* technique of [7], and most engineering applications have such constraints. Finally, to our knowledge, our hybrid approach (MADS with TGP) is the only one supported by publicly available software (see `NOMAD` v.3.6.2 [2,42], and Appendix C of the `NOMAD` documentation).

The paper is organized as follows. Section 2 describes the MADS algorithm and TGP. Section 3 presents the new algorithm, called MADS–TGP, which integrates TGP as a surrogate, and thereby derived statistical search criterion, into the MADS framework. Finally, Section 4 gives numerical results on a synthetic problem with many local optima, and also on two realistic blackbox applications. We conclude with a discussion in Section 5.

2 MADS and TGP

This section independently presents the MADS algorithm and the treed Gaussian processes. Emphasis is put on the flexibility of MADS allowing a convenient integration of TGP.

2.1 Mesh adaptive direct search (MADS)

The MADS algorithm is a directional direct search method introduced in [5,6], which generalizes the GPS method. The main advantage of MADS over GPS is the use of dense sets of directions instead of a finite number of fixed directions. As it is detailed below, this is possible thanks to the use of two different kinds of mesh size parameters Δ_k^m and Δ_k^p instead of the GPS single Δ_k mesh size parameter.

At the k^{th} iteration of the algorithm, each trial point lies on the mesh

$$M_k \;=\; \{x + \Delta_k^m Dz \;:\; x \in V_k,\; z \in \mathbb{N}^{n_D}\} \subset \mathbb{R}^n \tag{2.1}$$

where $V_k \subset \mathbb{R}^n$, or *the cache*, is the set of all points evaluated by the start of the iteration, $\Delta_k^m \in \mathbb{R}_+$ is the *mesh size parameter*, and D is a fixed matrix in $\mathbb{R}^{n \times n_D}$ composed of n_D columns representing directions. D must satisfy some conditions but typically corresponds to $[I_n \; -I_n]$ with I_n the identity matrix in dimension n, and $n_D = 2n$.

Each iteration is divided into two major steps, called *search* and *poll*. The search step allows for great flexibility and enhancements to the algorithm. It allows the generation of a finite number of trial points and the convergence, discussed at the end of the section, is ensured (i.e., unchanged) as long as these points lie on the mesh. The search can be problem-specific: a user with some insight about good designs can inject this knowledge with an appropriate search procedure. It may also be generic, for example a latin hypercube (LH) design [48] may be used in order to encourage exploration. More sophisticated examples include particle swarm [61], variable neighborhood search [4], and quadratic models [20,23].

The present work concerns the development of a search scheme based on TGP surrogates, and accompanying improvement and exploration oriented statistics.

The poll step ensures theoretical convergence. It constructs a set of candidates, P_k, called the *poll set*, defined as

$$P_k = \{x_k + \Delta_k^m d : d \in D_k\} \subseteq M_k,$$

where D_k is the set of *polling directions* constructed by taking combinations of the set of directions D. The way the set D_k is chosen depends on the MADS implementation. With OrthoMADS [3], for example, large angles between poll directions are avoided.

The *poll size parameter* Δ_k^p defines the maximum distance at which poll trial points are generated from the current iterate x_k, which is also called the *poll center*. In practice, Δ_k^p is of the same order as $\Delta_k^m||d||$ for $d \in D_k$. As the algorithm proceeds, the independent evolution of the mesh and poll size parameters is designed so that the set of poll directions becomes dense in the unit sphere, once normalized, meaning that potentially every direction can be explored. See Figure 1 for a simple example of trial points that could be generated during the search and poll steps.

After the search and the poll steps, a final update step is performed. It first determines if the iteration is a success or a failure and it updates the mesh and poll sizes accordingly. In the case of success, these sizes are potentially increased, and they are decreased otherwise. This decrease is not of the same rate for the two parameters: the mesh size is reduced faster than the poll size, which allows more and more possible directions as the mesh becomes thinner. In the unconstrained case, a success occurs when the objective is improved (via a simple decrease condition on the objective). With constraints, a filter-type algorithm called the *progressive barrier* (PB) [7] is used. It is based on the following *constraint violation function* inspired by the filter method of [25]:

$$h(x) \;=\; \begin{cases} \displaystyle\sum_{j\in J} (\max\{c_j(x), 0\})^2 & \text{if } \; x \in \mathcal{X}, \\ \infty & \text{otherwise.} \end{cases} \tag{2.2}$$

The function h is nonnegative and $x \in \Omega$ if and only if $h(x) = 0$. Moreover, x is in $\mathcal{X} \setminus \Omega$ if $0 < h(x) < \infty$. Note that this definition does not take into account hidden constraints. In practice, when such a constraint is met, h is put to ∞. During the update step, success or failure is determined by the f and h values of the evaluated points, and by considering the following *dominance* relation between two arbitrary points x, y in $\mathcal{X}$: y *dominates* x if and only if $h(y) \le h(x)$ and $f(y) \le f(x)$ with at least one of the inequalities being strict. Basically, an iteration is declared successful if a new non-dominated point is found.

Many stopping criteria may be considered to terminate the algorithm, such as a threshold on the mesh size parameter Δ_k^m, or, as dictated by many hard applied problems, one can stop the procedure when a budget of evaluations has been exhausted.

The MADS algorithm is presented schematically in Figure 2. Perhaps the most attractive feature of the algorithm is that it can be shown to globally converge to a solution satisfying local optimality conditions. The proof, which depends on the degree of smoothness of the objective and of the constraints, relies on the Clarke nonsmooth calculus [19]. The interested reader can consult [6] for details on the MADS convergence analysis. The convergence results hold for any custom search strategy, including the one presented here, as long as it generates a finite number of trial points on the mesh. Furthermore, since convergence is robust to the choice of optional searches [Step 1.1 in Figure 2], they can lend convergence to the otherwise heuristic searches of a more global flavor, e.g., using TGP and EI which we discuss in the following subsection.

Finally, note that with MADS, parallelization is possible and can be achieved by performing concurrent blackbox evaluations [43] or more sophistically by decomposition [8]. However since the focus of this work is the use of TGP surrogates inside the search step, parallelism is not discussed further.

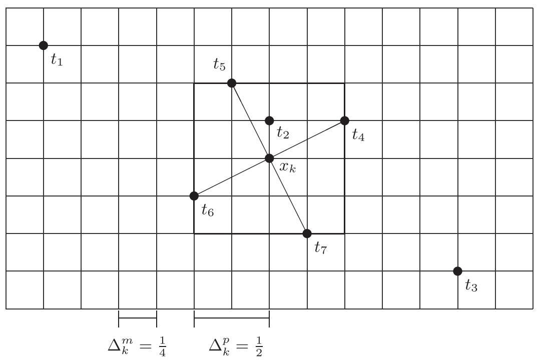

Figure 1: Example of MADS directions consistent with the ones defined in [3], in the case $n = 2$ and at iteration k of the algorithm. Thin lines represent the mesh of size Δ_k^m, and thick lines the points at distance Δ_k^p from x_k in the infinity norm. The poll trial points $P_k = \{t_4, t_5, t_6, t_7\}$ lie at the intersection of the thick and thin lines. After an iteration fails, Δ_k^m is reduced faster than Δ_k^p, and the number of possible locations grows larger. Some possible search set $S_k = \{t_1, t_2, t_3\}$ is also illustrated. Search points can be anywhere on the mesh as long as their number remains finite.

```
[0] Initialization
    starting point: $x_0 \in \mathcal{X}$
    initial mesh and poll size parameters: $\Delta_0^m$, $\Delta_0^p > 0$
    $k \leftarrow 0$
[1] Iteration $k$
    [1.1] SEARCH (optional)
        evaluate $f$ and $h$ on a finite set $S_k \subset M_k$
    [1.2] POLL (optional if SEARCH was successful)
        compute poll directions and trial points set $P_k$
        evaluate $f$ and $h$ on $P_k$
[2] Updates
    determine success or failure of iteration $k$
    update incumbent $x_{k+1}$
    update mesh and poll size parameters
      (reduce after failures)
    $k \leftarrow k + 1$
    goto [1] if no stopping condition is met
```

Figure 2: A simplified version of the MADS algorithm. See Figure 1 for some examples of search and poll points.

$\boxed{2.2}$ Treed Gaussian processes

One can view blackbox optimization as an example of sequential design of a computer experiment. In computer experiments, the "data" arise from expensive computer simulation, as realizations of the output of those simulations for a set of input configurations. Sometimes one assumes some limited knowledge of how the simulations work, but often the code execution remains opaque. The design and analysis of computer experiments is a rich field, see, e.g., [54]. However, their design for optimization (finding inputs that yield optimal/small outputs) has received rather less attention, until recently.

A stationary Gaussian process (GP) regression is the canonical statistical model for data arising from computer experiments. GPs represent one way of nonparametrically characterizing a zero-mean random process $Z(x)$ via its covariance $C(x, x') = \sigma^2 K(x, x')$. Let $Z_p = (z^1 \ z^2 \ \dots \ z^p)^T$ be observed responses (for one of the blackbox outputs, say f) at inputs $x^1, x^2, \dots, x^p$. Conditional on this data, the (posterior) predictive distribution of $Z(x)$ at a new input x is normal with

$$\begin{array}{rrcl} \text{mean} & \hat{z}_p(x) & = & k_p^T(x) K_p^{-1} Z_p, \\ \text{and variance} & \hat{\sigma}_p^2(x) & = & \sigma^2[K(x,x) - k_p^T(x) K_p^{-1} k_p(x)], \end{array} \tag{2.3}$$

where $k_p^T(x)$ is the p–vector whose i^{th} component is $K(x, x^i)$, and K_p is the $p \times p$ matrix with i, j element $K(x^i, x^j)$. These are sometimes called the *kriging equations*. The mean $\hat{z}_p(x)$ can be used as an emulator or surrogate for the expensive blackbox evaluations. Observe that a $p \times p$ inverse for K_p^{-1} is required, making the scheme require at least $\mathcal{O}(p^3)$ computing time. Once K_p^{-1} is obtained, each prediction at a new x requires $\mathcal{O}(p^2)$ time.

It is typical to specify $K(\cdot, \cdot)$ parametrically, e.g.,:

$$K_{d,g}(x, x') = \exp\left\{ -\frac{||x - x'||}{d} \right\} + g 1_{x=x'}$$

$$\text{or} \quad K_{d_1,\dots,d_n,g}(x, x') = \exp\left\{ -\sum_{j=1}^{n} \frac{(x_j - x'_j)^2}{d_j} \right\} + g 1_{x=x'},$$

which are known as the isotropic and separable *nugget*-augmented correlation functions, respectively. Both are supported by TGP, with the latter separable option as the default and posterior inference (given y-values) for the unknown parameters being carried out by Monte Carlo, again requiring $\mathcal{O}(p^3)$ time [more details below]. The d parameter(s), called the *characteristic lengthscale(s)*, govern the rate of decay of spatial correlation radially, or in each input direction. The *nugget* parameter, g, allows for measurement error or noise in the stochastic process. This causes the emulator to smooth rather than interpolate the responses, which is becoming widely recognized as a superior emulation tactic even when the outputs are observed deterministically, i.e., without noise [31]. For more examples of correlation functions popular in the spatial and computer experiments literatures, see, e.g., [1, 52].

Given a GP surrogate, a step in a search for the minimum of f may consider the *improvement* $I(x) = \max\{f_{\min} - Z(x), 0\}$. Here, $f_{\min}$ is either the smallest (valid) objective value found so far, or the global minimum of a GP surrogate predictive mean surface fit to the current cache of data. For a discussion of tradeoffs of these and other choices, see e.g., [30]. In particular, the next location can be chosen as $x^{p+1} \in \text{argmax}_{x \in \mathcal{X}} \mathbb{E}\{I(x)\}$, where $\mathcal{X}$ is the search domain of interest, often a bounding rectangle, but in the present work it is the set used in the definition of Ω in Problem (1.1). The expectation is over $Z(x)$, via Equation (2.3), and here we use $f_{\min} \equiv \min_{x \in \mathcal{X}} \hat{z}_p(x)$. This *expected improvement* (EI) is available analytically [40]:

$$\mathbb{E}\{I(x)\} = (f_{\min} - \hat{z}_p(x))\Phi\left(\frac{f_{\min} - \hat{z}_p(x)}{\hat{\sigma}_p(x)}\right) + \hat{\sigma}_p(x)\phi\left(\frac{f_{\min} - \hat{z}_p(x)}{\hat{\sigma}_p(x)}\right), \tag{2.4}$$

where Φ and ϕ are the cumulative distribution and probability density functions of a standard normal distribution, respectively. Basically, EI is the cumulative distribution of the predictive density that lies "underneath" $f_{\min}$. After $(x^{p+1}, z^{p+1} = f(x^{p+1}))$ is added into the design, and the surrogate model fit is updated based on all $p+1$ points, the procedure repeats. Given a fixed GP parameterization, the resulting iterative procedure is known to converge to a global minimum of f under certain conditions. For a review of related results for a family of similar methods, see [14].

There are many variations on this setup, particularly when the goal is exploration rather than optimization. The predictive variance $\hat{\sigma}_p^2(x)$ is a good heuristic for exploration. Another, more global, heuristic involves computing the average global (over the entire input space) reduction in predictive variance $\Delta\sigma^2(x)$ provided by the addition of a new input point x. This averaging requires (numerical) integration over the input space

$$\Delta\sigma^2(x) = \int_{\mathcal{X}} \left(\sigma_p^2(y) - \sigma_{p,x}^2(y)\right) dy. \tag{2.5}$$

The integrand calculates the reduction in variance at each point y in the input space after x is added in as the $(p+1)^{\mathrm{st}}$ point in the design. Here, $\sigma_p^2(y)$ represents the variance at y before x, and $\sigma_{p,x}(y)$ afterwards. The former follows Equation (2.3), and the latter requires an adjustment for the correlation between x and y. Crucially, this calculation is analytic for GP (and linear) models, e.g., [29, 55, 56]. As so-called *active learning* heuristics for GP models, their first use was [55]; in the statistics community these are more generically labeled as *sequential design* heuristics.

GPs are popular for such tasks because they enjoy theoretical properties which allow for proofs about convergence of the sequential procedure (solving certain optimization problems), and because they show favorable out-of-sample predictive performance relative to other modern nonparametric regression methods. But they are not without disadvantages. The most important practical problem is that GP inference requires $\mathcal{O}(p^3)$ computation. Another problem is the typical simplifying stationary assumption, which asserts that the covariance structure has the same properties throughout the entire input space. A consequence of this is that predictive uncertainty exhibits the same characteristics uniformly over the input space. This is most readily appreciated through Equation (2.3), where we see that $\hat{\sigma}_p^2(x)$ does not depend on the responses Z_p. In many real-world applications, like optimization, such a uniform modeling of uncertainty will not be desirable. Instead, some regions of the input space may exhibit larger predictive uncertainty than others.

These shortcomings, in whole or in part, can be addressed by techniques such as the ones described in [10, 11, 51]. In the present paper we consider a surrogate model which can overcome these limitations by partitioning the input space into disjoint regions, wherein independent stationary GP models are fit. Partitioning represents a straightforward mechanism for creating a nonstationary model, and a computationally thrifty one by taking a divide-and-conquer approach. The most tractable partitioning scheme is recursive axis-aligned splits as provided by trees, leading to the so-called treed GP (TGP) model [28]. TGP is basically a generalization of the Bayesian CART (classification and regression tree) model [17], using GPs at the leaves of the tree instead of the usual constant values or linear regressions [18]. A concern about trees is that they lead to discontinuous surrogate model fits. However, Bayesian model averaging over the joint tree and leaf-model (GP) posterior distribution by Markov chain Monte Carlo (MCMC) tends to yield mean fitted functions that are quite smooth in practice, especially when GPs are used at the leaves, giving fits that are indistinguishable from continuous functions except when the data call for the contrary [28].

Beyond computational and modeling flexibility, two key aspects of the TGP model are attractive in the direct optimization context. One is software. Its optimized C implementation is easily extracted from the open source tgp package [27] for R [58]. The other is that this implementation includes EI calculations for optimization, and $\Delta\sigma^2(x)$ calculations for exploration, as well as other active learning heuristics. As the GP and CART models are special cases of the TGP model (i.e., with no treed partitioning, or with constant/linear leaf models, respectively) the package represents an omnibus toolset for statistical optimization, and thus is ripe for pairing with a direct optimization method, like MADS. In the Bayesian model averaging setup, it is harder to get convergence guarantees for optimization. But with an appropriate pairing with a direct blackbox optimizer [as we describe], its convergence properties can be borrowed for the whole procedure. This approach has been taken to combine TGP/EI with direct search [57] via the APPS optimizer [34] which leveraged a generalized exponentiated improvement statistic to obtain a more global search. It also employed a thrifty algorithm for ordering the candidates which was crucial to synchronizing the two methods while minimizing computational effort. Both are supported by the tgp software (see [33], Section 4).

$\boxed{3}$ Incorporation of TGP into the MADS framework

TGP is used twice in the MADS framework of Figure 2. The augmented algorithm is called MADS–TGP, featuring *TGP search* and *TGP ordering* stages. A series of short sections (from 3.1 to 3.5) covers the specifics, and Figure 3 gives a high- level description. One of the key points of the TGP search is the combined use of several techniques in order to generate candidates, comprised of classical oracle points provided by surrogate optimization, and points identified by statistical criteria. This is detailed in Section 3.4. The TGP ordering, occurring during the poll step, is described in Section 3.5. Note that TGP is used inside the MADS framework and it leaves the constraint handling to the progressive barrier. Since this technique has been designed for general constraints, MADS–TGP automatically inherits this robustness and should be unaffected by the kind of constraints that are used.

At each iteration k, separate TGP models are fit, independently, for the objective and each constraint. These surrogates are denoted $f^s \simeq f$ and $c_j^s \simeq c_j$, $j \in J$. A single, combined, surrogate for the constraint violations, denoted $h^s \simeq h$, is then obtained using Equation (2.2) by replacing c_j with their surrogates c_j^s, $j \in J$. Once the surrogates have been fit to the data, their predictive distributions may be used in a variety of ways. They can be used in the traditional way: as an emulation or replacement for expensive blackbox evaluations of the objective and/or the constraints. Or, they can be used to extract active learning statistics like the expected improvement (for optimization), or the expected reduction in variance (for exploration).

The TGP models must be refit from scratch at the beginning of each iteration, assuming that new data has been added to the cache. Unfortunately, the implementation (in tgp) does not allow for a fast update of the fits in light of the new data. This is because the tree aspect requires MCMC inference which is not easily updated when the data are augmented. If only GP surrogates are used, then it may be possible to obtain a thrifty update in light of new data [32], but we do not consider that here because we find that partitioning is essential for many real-world applications including the ones we discuss in Section 4.

The implication is that the construction of (possibly several) surrogate TGP models may be computationally demanding, and thus time consuming. This is not an issue when dealing with costly blackbox simulations, that can take hours or even days to complete, as for example in [12, 46]. By comparison, TGP fits will be relatively swift.

```
[0] Initialization
[1] Iteration $k$
    [1.1] TGP SEARCH
          select data points $X$ and $Y$ from $V_k$ (Section 3.1)
          choose prediction points $\tilde{X} \subseteq M_k$ (Section 3.2)
          for each output $j \in \{1, 2, \ldots, m+1\}$
              $\left(\tilde{V}^j, \tilde{E}^j, \tilde{Y}^j\right) \leftarrow$ construct($X, Y^j, \tilde{X}$)
          surrogate available: $f^s$, $c^s_j$ ($j \in J$), and $h^s$ can be
            computed with the predict function
          surrogate optimization (Section 3.3)
              solve $\min_{x \in \Omega^s} f^s(x)$
              with $\Omega^s = \{x \in \mathcal{X} : c^s_j(x) \leq 0, j \in J\}$
              denote $x^F$ and $x^I$ the solutions
              project $x^F$ and $x^I$ to $M_k$
          construct $S_k \subseteq \{\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{\tilde{p}}, x^F, x^I\}$ (Section 3.4)
          evaluate opportunistically $f$ and $h$ on $S_k$
    [1.2] POLL (optional if the SEARCH was successful)
          compute $P_k$
          sort points of $P_k$ accordingly to $f^s$ and $h^s$ (Section 3.5)
          evaluate opportunistically $f$ and $h$ on $P_k$
[2] Updates
```

Figure 3: High level description of the MADS–TGP algorithm. Only the differences compared to the algorithm from Figure 2 are reported. The `construct` and `predict` functions symbolize the calls to the `tgp` package necessary to construct the models and to obtain the predictions at some given locations. In particular, the `predict` function evaluates Equations (2.3–2.5) for each MCMC sample from the TGP model. Other details, definitions and use of X, Y, $\tilde{X}$, $\tilde{V}$, $\tilde{E}$, and $\tilde{Y}$, are given in Sections 3.1 to 3.5. This version only includes the TGP search but other types of searches may still be integrated.

3.1 Data points

The data points that are used for the surrogate model (TGP) fitting are selected from the cache V_k introduced with the mesh definition of Equation (2.1). This mathematical set only describes the evaluated points, not the corresponding blackbox output values, but, in a practical implementation, these are available from the cache data structure. Only points in $\mathcal{X}$ that do not violate a hidden constraint are considered, as these are the only ones for which output values are available. In the present version of the method, no mechanism is used in order to measure the probability that a trial point violates a hidden constraint or is outside $\mathcal{X}$ in the case where $\mathcal{X}$ is not just defined by bounds. In future work, we plan to explore the use of an additional surrogate model to estimate the probability of the violation of this type of constraint [30].

Data points are denoted by $X = \left[x^1\ x^2\ \dots\ x^p\right]^T \in \mathbb{R}^{p\times n}$. The corresponding output values are represented by $Y \in \mathbb{R}^{p\times(m+1)}$. The first m columns of this matrix correspond to the constraints and the last column to the objective:

$$Y = \begin{bmatrix} c_1(x^1) & c_2(x^1) & \dots & c_m(x^1) & f(x^1) \\ c_1(x^2) & c_2(x^2) & \dots & c_m(x^2) & f(x^2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_1(x^p) & c_2(x^p) & \dots & c_m(x^p) & f(x^p) \end{bmatrix} \in \mathbb{R}^{p\times(m+1)}.$$

The number of data points must be at least $n+1$, otherwise the TGP model cannot be fit as there will be more degrees of freedom than the number of data points [28]. If this number cannot be reached, then no surrogate is constructed at iteration k, i.e., the search is aborted.

A limit $p_{max} > n+1$ on the maximum number of data points is also considered. If more than p_{max} cache candidates are available, then the p_{max} candidates that lie closest in norm to the current iterate x_k are selected to build the surrogates. The following heuristic is applied for choosing this limit in practice:

$$p_{max} = \max\left\{30, \min\left\{100, \lfloor\sqrt{180n}\rfloor\right\}\right\} = \begin{cases} 30 & \text{if } 1 \leq n \leq 5, \\ \lfloor\sqrt{180n}\rfloor & \text{if } 6 \leq n \leq 56, \\ 100 & \text{otherwise.} \end{cases}$$

Although superficially arbitrary, these choices have been carefully made in order to strike a balance between computational time and model quality, especially regarding the local v.s. global nature of the model. For example, when $p \gg p_{\max}$ the TGP model *could* take on a more local flavor if the most recent $p_{\max}$ points are heavily concentrated in one part of the input space. Note that cases with $n > 50$ are suggested by our rule but that in practice we rarely consider such situations in DFO.

3.2 Prediction points

Prediction points are denoted by $\tilde{X} = \left[\tilde{x}^1\ \tilde{x}^2\ \dots\ \tilde{x}^{\tilde{p}}\right]^T \in \mathbb{R}^{\tilde{p}\times n}$ and correspond to the locations where `tgp`, during the surrogate construction for an output $j \in J \cup \{m+1\}$, will simultaneously provide samples from the full (model-averaged) posterior predictive distribution. In turn, these samples can be used to calculate the active learning heuristics: the expected reduction in variance $\Delta\sigma^2(x)$ of Equation (2.5), denoted by $\tilde{V}^j \in \mathbb{R}^{\tilde{p}}$, and the EI of Equation (2.4), denoted by $\tilde{E}^j \in \mathbb{R}^{\tilde{p}}$.

As described in Section 3.4, these points are entertained as candidates for true blackbox function evaluations which are chosen on the basis of the active learning heuristics, on which we shall say more in Section 3.5. In addition, the surrogate values (i.e., predictive means) for the objective and the constraints are predicted at each of the prediction points. The result of these predictions is stored in the matrix $\tilde{Y} \in \mathbb{R}^{\tilde{p}\times(m+1)}$ which shares the same structure as the Y matrix: the first m columns represent the constraints, and the last column represents the objective.

In order to ensure convergence, the prediction points must lie on the mesh M_k established at iteration k. Therefore, projections of LH samples onto the mesh are used to generate $\tilde{X}$. We impose a limit of 500 such points. This limit may not be reached because there may be fewer than 500 mesh locations inside the bounds in the initial stages of the algorithm, and in that case we consider all of the mesh points.

$\boxed{3.3}$ Optimization with the surrogate

When the TGP surrogate requires optimizing, i.e., as per the traditional use of a surrogate model, we use MADS. Since TGP predictive surfaces are smooth, other optimization methods may be similarly well-suited to this task, and these may yield a faster convergence rate. However, it seems simplest to stay within the MADS framework from an implementation perspective (i.e., not incorporating yet another method). Since MADS theory ensures convergence to a solution satisfying some local conditions, this step is not the computational bottleneck of the method. So the replacement of MADS by a derivative-based algorithm, say, is not expected to yield dramatic improvements.

The optimization domain for the surrogate is defined by the smallest rectangle containing all of the data points of X. The starting point of the optimization is one of the prediction points, i.e., one of the rows of $\tilde{X}$. This point, denoted by $\tilde{x}^i$, $i \in \{1, 2, \dots, \tilde{p}\}$, is chosen so that it is predicted to be the best feasible point. More precisely, the index i satisfies $i \in \underset{k \in \{1,2,\dots,\tilde{p}\}}{\operatorname{argmin}} \tilde{Y}_k^{m+1}$ subject to $\tilde{Y}_k^j \leq 0$, for all $j \in J$. If the index i is impossible to determine because there is no row in $\tilde{Y}$ with all first m components being negative, then i is chosen so that $i \in \underset{k \in \{1,2,\dots,\tilde{p}\}}{\operatorname{argmin}} \sum_{j=1}^{m} \left(\tilde{Y}_k^j\right)^2$.

We allow the optimization of the TGP surrogate to consider locations outside the mesh. If the optimization completes, the solutions x^F and x^I are gathered. These points correspond to the feasible point with the smallest f value and the infeasible point with the smallest h value, respectively. Before being proposed as candidates for the true functions, x^F and x^I are projected to the mesh and we call them the *oracle points*.

$\boxed{3.4}$ Selection of the search points

The TGP search generates mesh candidates to be evaluated. The present section explains how to pick a limited number of candidates, and how to determine the priority in which they should be evaluated by the blackbox and subsequently added to the cache.

Each search step populates three lists of potential candidates for blackbox evaluations based on information provided by the TGP model predictive distribution. Two of the three lists involve the prediction points of $\tilde{X}$. The first one is derived from EI calculations (see Equation (2.4)), which are provided by `tgp` (via the argument `improv=TRUE`) as the matrix $\tilde{E} \in \mathbb{N}^{\tilde{p} \times (m+1)}$ including ranking information for serial inclusion. The ranking is related to, but not identical to, the ordering of EI values at the candidates. Rather, the ordering considers how the EI calculations update in light of the inclusion of higher ranked points, therefore emphasizing exploration as the list is considered in rank order. For more details, see Section 3 of [33]. The list of potential candidates is constructed by first filtering the points that are predicted to be infeasible using the first m columns of $\tilde{Y}$, and then by considering the rankings for the objective function in the $(m+1)^{\text{th}}$ column of $\tilde{E}$. Note that in practice the generation of the expected improvement is disabled for the first m outputs and the corresponding columns of $\tilde{E}$ are ignored.

The second list is populated by evaluations of the expected reduction in variance from Equation (2.5) at the $\tilde{X}$ locations, obtained when providing the argument `ds2x=TRUE` to `tgp`. This yields the matrix $\tilde{V} \in \mathbb{R}^{\tilde{p} \times (m+1)}$. For each output $j \in J \cup \{m+1\}$, the prediction point with the greatest value in $\tilde{V}^j$ is selected. The second list of candidates contains then at most $m+1$ points that are expected to be good locations in order to improve the surrogate quality.

Finally, the third list of potential candidates is the list of oracle points x^F and x^I obtained from the surrogate optimization described in Section 3.3. The feasible oracle point x^F is given priority over the infeasible oracle point x^I.

Obtaining the final subset of candidates $S_k \subseteq \tilde{X}$ for blackbox evaluation—the list of TGP search points for iteration k—proceeds by round-robin selection from each of the three lists in turn, Since we believe that the three lists are complementary and that no one should be preferred to the other. First, the highest ranked candidate is taken from the expected improvement list, then the oracle list, and finally from the expected reduction in variance list. This is repeated until a limit on the maximum number of trial points is reached. This limit is fixed to 10 points for most of the numerical tests of Section 4, and is compared with the values 2 and 5 in the same section (see Table 10 and Figure 12).

$\boxed{3.5}$ TGP ordering for a list of evaluations

This section relates to the second use of TGP, the TGP ordering strategy, which is not related to the TGP search described in Sections 3.3 and 3.4. The ordering occurs during the poll step of the MADS–TGP algorithm [Step 1.2 in Figure 3] while the search occurs during the search step [Step 1.1 in Figure 3]. Ordering consists of using surrogates to give a priority to the candidates before their "true" evaluation. It takes place during the poll, but it could also be used for any sequence of evaluations, for example, from an additional search step. It exploits the opportunistic strategy which consists of interrupting a sequence of evaluations as soon as a success has been achieved. Our experience in DFO as well as a recent study [50] indicate that in general the opportunistic strategy performs better than the complete strategy, which consists of waiting for all evaluations to terminate.

TGP ordering consists of sorting the poll points according to their predictive f and h values, as provided by the `predict.tgp` function. More specifically, points are compared in a filter-type manner: if a point has better values for *both* f and h, then it is given priority. If that is not the case, we consider the *last successful direction* defined by the last two iterates of the algorithm. We then give priority to the candidates having the smallest angles with the last successful direction. The ordering strategy is comparable to the ones based on quadratic models in [20, 23] or simplex gradients in [24], except that TGP surrogates are used, and last successful directions are considered instead of simplex gradients.

Finally note that, for the sake of a simpler implementation, the surrogate used is the one created during the search step, and that this surrogate does not include the evaluations that occurred after the surrogate construction. These evaluations will be included in the surrogate models of the next iteration.

$\boxed{4}$ Numerical results

The implementation for our numerical tests is based on the `NOMAD` software [2, 42] for the MADS part and on the `tgp` package [27] for the TGP surrogates. The MADS–TGP algorithm is compared with MADS alone and MADS with the quadratic models [20]. Five test cases are considered. Two synthetic problems and three realistic blackbox applications representing the main class of problems for which MADS–TGP is designed. The objective of these tests is to measure the impact of TGP as a "copilot" for MADS. For this reason, no tuning is performed on the `NOMAD` parameters which are left at their recommended default values – except for the model search and the evaluation ordering strategies which are being tested here.

Both features are included in the December 2013 release of `NOMAD`, version 3.6.2.

For the five problems and the three methods, ten independent runs were made, each with a different random seed. For each execution, the stopping criteria are a maximum of 1,000 blackbox evaluations and a minimum mesh size of 1E−13 (this the `NOMAD` precision).

The starting point $x_0 \in \mathcal{X}$ is taken as the best solution from a LH sampling of 20 points, except for STYRENE for which 10 more LH points are used for a total of 30, to account for its larger dimension. These LH evaluations are included in the total account of blackbox evaluations.

Tables and graphs summarize all these executions. In the different tables, the columns "MADS alone" and "Quadratic" correspond to the algorithms that are not using TGP, while the "TGP" column indicates the new method MADS–TGP. The "start" column contains the best value achieved after the initial LH evaluations. Sometimes, this initial sampling was unable to provide a feasible solution. In these cases, the "−" mark is used. For the analytic problems, convergence is often reached before 1,000 evaluations. In this case, the "eval." columns indicates the numbers of evaluations used in total. Otherwise, the limit of 1,000 evaluations was always reached before convergence and thus the "eval." column is not reported. We show instead a column labeled as "impr." which shows the relative improvement compared to the initial solution, in %, when this initial solution is feasible. Finally, the "obj." column shows the value of the objective function when the algorithm stopped. For each row, the entries in bold correspond to the best objective values, and the last row of the table indicates the average values over the ten runs.

The graphs used to illustrate the comparisons are *performance* and *data profiles* for the ten executions from different starting points. These profiles were first introduced in the DFO context in [49]. In these figures, each curve corresponds to one method: MADS alone, Quadratic, and TGP. The y-axis is the same for the two types of profiles and it indicates the proportion of problems *solved*. Here, the term "solved" means that the different solution values are compared against the best values obtained amongst all executions, not necessarily the optimal or the best known values. In addition, each comparison is a relative one, with reference to a certain precision, except for the first analytic problem (GRIEWANK), because the optimal value of zero has been obtained by several executions. The x-axis of the performance profiles specifies that precision, and only the final solutions of each executions are considered. For the data profiles, the precision is typically set to 0.1, and the x-axis corresponds to the number of evaluations. To summarize, performance profiles illustrate the quality of solution of the methods for different precision values, and data profiles illustrate the convergence for given a fixed precision.

4.1 Analytic problems

This first part of the numerical tests focuses on two analytic problems. One small, bound-constrained problem with many local optima, and one that can be considered a hybrid between an academic and a realistic application.

4.1.1 Synthetic data with many local optima

We consider the GRIEWANK function [35] as a typical example of a simple function exhibiting many local optima, which is one of the targeted application of the MADS–TGP method. It has $n = 2$ variables and is expressed by:

$$f(x_1, x_2) = 1 + \sum_{i=1}^{2} \frac{x_i^2}{4000} - \prod_{i=1}^{2} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

with $-600 \leq x_1, x_2 \leq 600$. The true global optimum is located at $(0,0)^T$ with $f(0,0) = 0$. Note that this function is not a blackbox but is used here anyway in order to illustrate the global optimization capabilities of the MADS–TGP algorithm. A graphical representation of the function showing its many local optima is given in Figure 4.

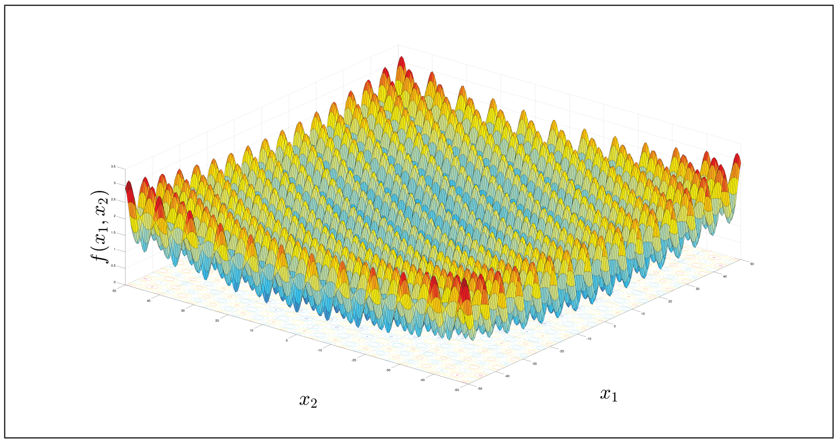

Figure 4: Representation of the GRIEWANK function in which many local optima appear.

Results for the GRIEWANK function are summarized in Table 1. The first thing that we notice is that convergence was always reached before 1,000 evaluations and that the executions stopped earlier for the runs without TGP. However, this does not mean that using TGP required more evaluations in order to achieve the same quality of solution. Rather, using surrogates generally yielded better solutions sooner. The fact that MADS without TGP stopped sooner is indicative of the fact that it was easily becoming stuck at local minima.

Now we focus on the quality of solution. Clear improvements were achieved by quadratic models over MADS alone, probably due to the smoothness of the function. Using TGP instead yields even better results. Figure 5 presents the performance (left part of the figure) and data (right part) profiles for the ten executions from different starting points. Here, because the optimal value of zero has been obtained by several executions, this precision corresponds to the absolute values of the various solutions. The profiles show that MADS–TGP and MADS with quadratic models could solve one instance out of 10 (x-axis for a null precision), but that MADS–TGP always gave a solution strictly under 0.1 (x-axis for precision $\geq$ 0.1). This can be also observed in Table 1. Finally, the left subfigure shows that TGP clearly dominates the two other methods because the associated plot is always on top of the two other plots. Similarly, the performance profile shows that MADS with quadratic models performs better than MADS alone, probably due to the smooth nature of the function. The data profile in the same figure shows that with fewer than 100 evaluations, MADS alone and MADS with quadratic models perform better. However, after 100 evaluations, MADS–TGP is much more efficient. Note that using other precisions for the data profile does not change this hierarchy, but 0.1 allows a better representation.

We conclude our study of the GRIEWANK function by indicating that the MADS–

Note that although objective values have been rounded to 4 decimals for display reasons, the zero values that may appear are the actual values displayed by `NOMAD` (with a precision of 1E-13). No value between 0 and 0.0074 was encountered.

Table 1: Results for the GRIEWANK problem ($n = 2$, $m = 0$).

#	start	MADS alone		Quadratic		TGP	
		eval.	obj.	eval.	obj.	eval.	obj.
G01	147.45	366	2.4553	364	0.5598	**693**	**0.0296**
G02	63.85	341	0.1454	**355**	**0**	540	0
G03	79.43	429	1.4868	**376**	**0.0666**	598	0.0666
G04	81.88	385	0.3428	355	0.2663	**615**	**0.0074**
G05	148.78	371	0.0395	389	0.1183	**580**	**0.0271**
G06	12.87	382	0.7965	379	0.2367	**564**	**0.0271**
G07	9.95	407	0.1849	381	0.1849	**603**	**0.0395**
G08	102.19	418	1.3092	419	0.1257	**614**	**0.0271**
G09	58.48	443	0.0666	**334**	**0.0074**	642	0.0296
G10	38.61	388	0.2169	400	0.0197	**651**	**0.0074**
avg	74.35	393	0.7044	375	0.1586	**610**	**0.0261**

TGP version takes approximately 10 minutes to complete an optimization on a 2010 laptop computer. While the executions of MADS alone and MADS with quadratic models are almost instantaneous, the gain in terms of quality of the solution is worth the additional cost of the TGP surrogates.

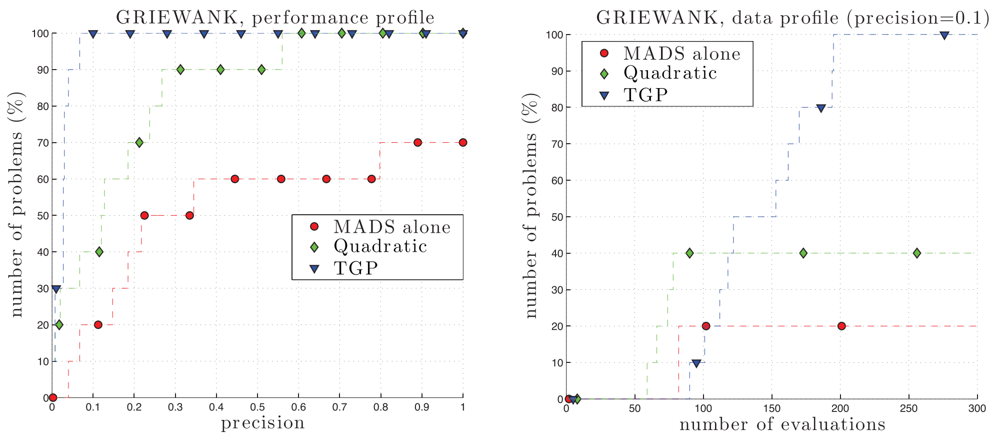

Figure 5: Performance (left) and data (right) profiles for the GRIEWANK problem ($n = 2$, $m = 0$). In the data profile, the plots stop at 300 evaluations because there was no evolution after that point.

4.1.2 Instance from the CUTEst library

This section considers an example taken from the CUTEst [26] library, often used in nonlinear optimization. A few examples from this collection of analytic problems refer to real-world applications and/or to irregular functions for which no derivatives are available. Amongst these problems, we selected the HS67 instance as the only one implying a numerical method and with a reasonable dimension for any currently available derivative-free solver, i.e. less than 20 variables and 20 inequality constraints. The HS67 problem is described in [38]. It has 3 variables, described in Table 2, and 14 constraints.

Variable	Lower bound	Best known	Upper bound
x_1	1E-5	1,728	2,000
x_2	1E-5	16,000	16,000
x_3	1E-5	98.13	120
Objective value	infeasible	-1,162.036326	infeasible

Table 2: Description of the HS67 problem variables. The coordinates of the best known solution have been rounded.

The objective function is $f(x_1, x_2, x_3) = -0.063y_2y_5 + 5.04x_1 + 3.36y_3 + 0.035x_2 + 10x_3$, in which the *coupling variables* $y_2, y_3, \ldots, y_8$ appear. These seven quantities are defined in the following system:

$$\begin{array}{rll}
0 \leq & y_2 = 0.01x_1(112 + 13.167y_6 - 0.6667y_6^2) & \leq 5{,}000 \\
0 \leq & y_3 = 1.22y_2 - x_1 & \leq 2{,}000 \\
85 \leq & y_4 = \frac{98000x_3}{y_2y_7+1000x_3} & \leq 93 \\
90 \leq & y_5 = 86.35 + 1.098y_6 - 0.038y_6^2 + 0.325(y_4 - 89) & \leq 95 \\
3 \leq & y_6 = \frac{x_2+y_3}{x1} & \leq 12 \\
0.01 \leq & y_7 = 35.82 - 0.222y_8 & \leq 4 \\
145 \leq & y_8 = 3y_5 - 133 & \leq 162
\end{array} \tag{4.1}$$

System (4.1) is solved using a numerical method based on two successive fixed-point loops, as illustrated by the pseudocode of Figure 6. The bounds on the 7 coupling variables define the 14 constraints of the problem.

```
y2 = 1.6 * x1;

do {
  y3 = 1.22 * y2 - x1;
  y6 = (x2+y3)/x1;
  tmp = 0.01*x1*(112+13.167*y6-0.6667*y6*y6);
  if ( abs(tmp-y2) <= 1E-4 )
    break;
  y2 = tmp;
}

y4 = 93.0;

do {
  y5 = 86.35+1.098*y6-0.038*y6*y6+0.325*(y4-89);
  y8 = 3*y5-133;
  y7 = 35.82-0.222*y8;
  tmp = 98000*x3/(y2*y7+1000*x3);
  if ( abs(tmp-y4) <= 1E-4 )
    break;
  y4 = tmp;
}
```

Figure 6: Fixed-point numerical methods used to compute the coupling variables y_2, y_3, ..., y_8 for the HS67 problem.

Results for the HS67 problem are presented in Table 3 and in Figure 7. Given a budget of 1,000 evaluations, all three methods provided a solution close to the best known value using a precision of 2.5E−3 For this reason, the values in Table 3 are shown with 6 decimals,

Table 3: Results for the HS67 problem ($n = 3$, $m = 14$). The "–" marks are used when the initial LH search could not find a feasible solution.

#	start	MADS alone		Quadratic		TGP	
		eval.	obj.	eval.	obj.	eval.	obj.
H01	–	843	-1,162.034269	1,000	-1,162.036273	**1,000**	**-1,162.036326**
H02	-243.627829	746	-1,162.035693	994	-1,162.036095	**1,000**	**-1,162.036326**
H03	-303.607164	893	-1,162.036200	991	-1,162.033948	**1,000**	**-1,162.036326**
H04	-1,111.032739	834	-1,162.036012	943	-1,162.036318	**968**	**-1,162.036326**
H05	-85.969838	713	-1,162.031528	901	-1,162.036291	**1,000**	**-1,162.036326**
H06	-688.425235	697	-1,162.035892	727	-1,162.035973	**1,000**	**-1,162.036326**
H07	–	797	-1,162.036104	1,000	-1,162.036325	**1,000**	**-1,162.036326**
H08	-541.370728	708	-1,162.035907	936	-1,162.035349	**1,000**	**-1,162.036326**
H09	–	894	-1,162.035889	928	-1,162.036307	**1,000**	**-1,162.036326**
H10	-623.856175	867	-1,162.035762	823	-1,162.036102	**1,000**	**-1,162.036326**
avg	-513.984244	799	-1,162.035326	924	-1,162.035898	**997**	**-1,162.036326**

and the plots in Figure 7 do not include a performance profile but rather three data profiles using the precisions 0.1, 1E-3, and 1E-7.

These profiles clearly show that, for precisions of 0.1 and 1E-3, the three methods behave approximately the same, but the high-precision data profile clearly shows the MADS–TGP dominance. It solved all instances while MADS alone and with quadratic models could only solve half the instances and MADS alone was never able to give the best known solution. In all cases, MADS–TGP took more evaluations than the other methods, but the data profiles demonstrate that these additional evaluations are not the reason why MADS–TGP dominates: after 800 evaluations, all the instances were solved by MADS-TGP, with a precision of 1E-7.

4.2 Tests on three realistic blackbox applications

This section first describes the three realistic applications on which the new method has been tested, and then presents the results. Note that the STYRENE and MDO applications are publicly available at S. Le Digabel's homepage [`https://www.gerad.ca/Sebastien.Le.Digabel/Publications`].

4.2.1 Description of the STYRENE problem

We first consider a chemical engineering simulator for styrene production described in [4] and studied in [9]. We refer to this application as the STYRENE problem. It has n =8 variables and m =11 constraints and the best known value is reported at -33,539,100 in [4], Table 5.

The blackbox simulates the chemical process schematized in the flowsheet of Figure 8, based on different logical blocks. Given the 8 input variables described in Table 4, it computes the values necessary to express the objective function and the constraints, using the following numerical methods: Runge-Kutta, Newton, fixed-point, secant, bisection, and other chemical engineering related solvers.

The objective represents the Net Present Value (NPV) of the process:

$$f(x) = \sum_{y=0}^{Y} \frac{(S_y - C_y)(1 - T_a) - I_y + D_y}{(1 + T_r)^y} ,$$

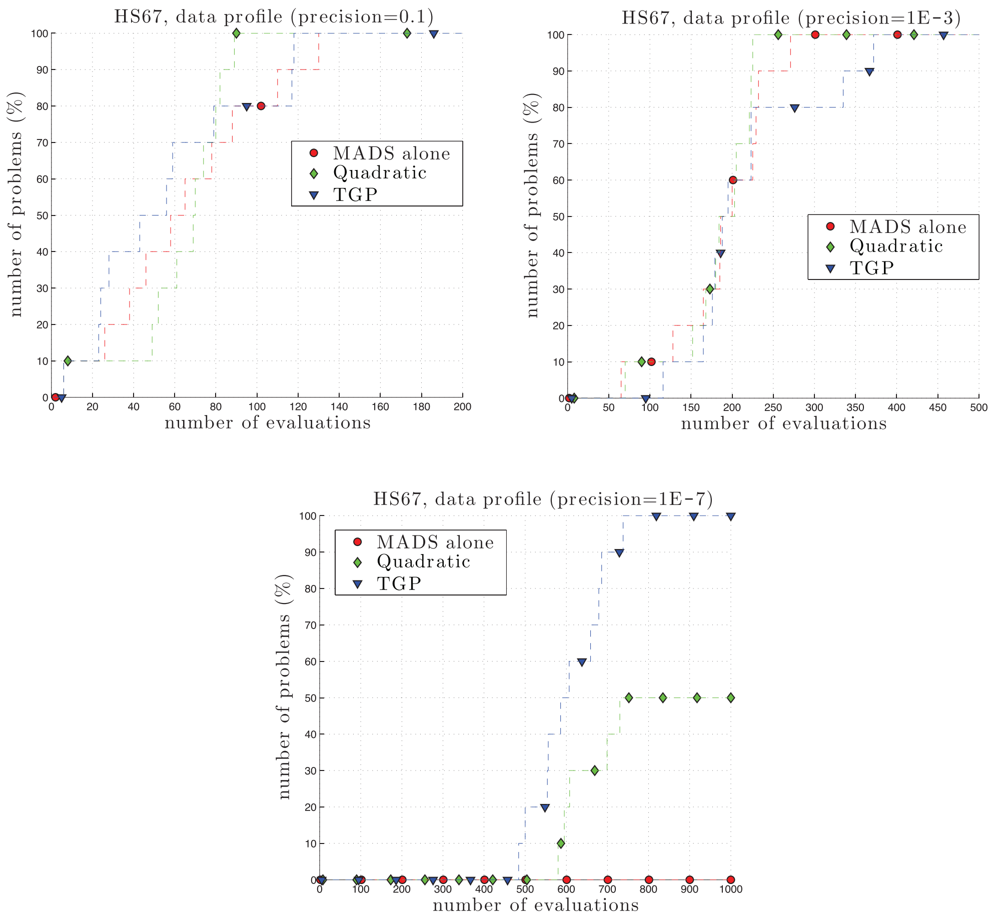

Figure 7: Data profiles for the HS67 problem ($n = 3$, $m = 14$), using three different precisions. In the data profiles, the plots are stopped when there was no more evolution.

where y denotes a year between 0 and Y, and the sales S_y, the operating costs C_y, the investment I_y, and the depreciation D_y, are direct outputs from the simulator. The constants T_a =0.4 and T_r =0.1 correspond to the income tax rate and the actualization rate, respectively.

The 11 constraints include requirements on the structural configuration of the columns SEP-STY and SEP-BZ, conditions if the mixture in block FIRE can burn, regulations on CO and NOX emissions, minimal purity of produced styrene and benzene, minimal overall ethylbenzene conversion into styrene, maximal payout time, minimal discounted cash-flow rate of return, maximal total investment, and maximal annual equivalent cost.

In STYRENE, hidden constraints are omnipresent: a representative 100 evaluations made on randomly sampled points yielded 57 failures. We note, however, that during an optimization process generating more physically realistic configurations, this rate drops to approximately 20%. Since a hidden constraint violation usually occurs in the middle of the simulation, it counts as one function evaluation in our results of Section 4.2.4.

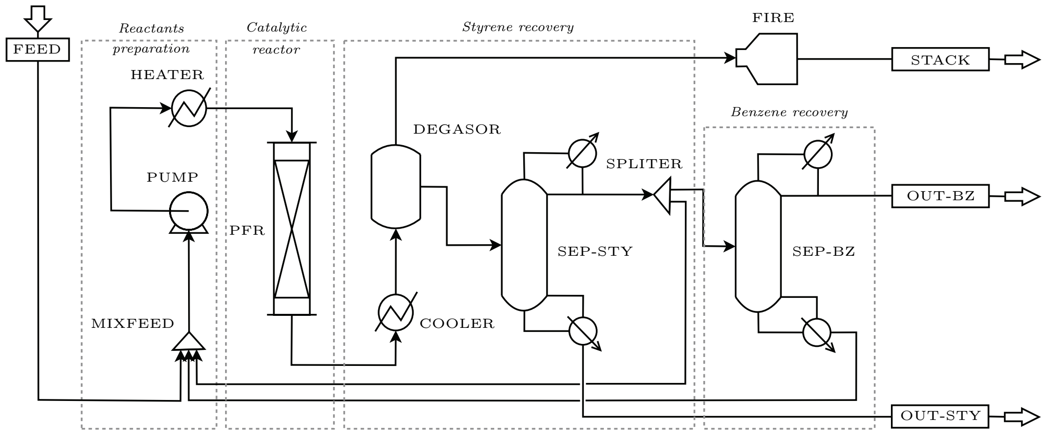

Figure 8: Flowsheet of the styrene production process. Taken from [4].

Variable description	Lower bound	Best known	Upper bound
Outlet temperature in block HEATER (K)	600	1,100	1,100
Length of reactor in block PFR (m)	2	16.98	20
Light key fraction in block SEP-STY	1E−4	0.09683	0.1
Light key fraction in block SEP-BZ	1E−4	1E−4	0.1
Outlet pressure of block PUMP (atm)	2	2	20
Split fraction in block SPLITER	0.01	0.2247	0.5
Air excess fraction in block FIRE	0.1	1.963	5
Cooling temperature of block COOLER (K)	300	403.0	500
Objective value	failure	-33,539,100	failure

Table 4: Description of the STYRENE problem variables. The coordinates and the value of the best known solution have been rounded.

4.2.2 Description of the LOCKWOOD problem

The second application is called the LOCKWOOD problem [47] and seeks the optimal extraction rates for six (fixed-locale) decontamination wells at a polluted site in Montana. This problem has $n = 6$ variables which represent the pumping rate for each well, in m^3/day. Their bounds and their best known values are described in Table 5. The best objective value in [47] is 23,714. The present work reports a solution with value 22,807.10.

The function to minimize is linear. It is the total pumping required to run the system, $f(x) = \sum_{j=1}^{6} x_j$, which is proportional to the cost of operating the system. The linear form of the objective is not exploited in this work.

The difficulty in this application is that the $m = 2$ constraints are two outputs c_1 and c_2 given by the Bluebird simulator [22], based on an analytic element method groundwater model. These outputs represent the amount of contaminant entering the Yellowstone river at the western side of the site, and exiting at the southern and eastern boundaries. The constraints are expressed as $c_j \leq$ 1E−3 for $j \in \{1, 2\}$.

Variable	Lower bound	Best known	Upper bound
x_1 (m^3/day)	0	341.80	20,000
x_2 (m^3/day)	0	5,275.25	20,000
x_3 (m^3/day)	0	13,566.49	20,000
x_4 (m^3/day)	0	2,160.32	20,000
x_5 (m^3/day)	0	777.27	20,000
x_6 (m^3/day)	0	685.98	20,000
Objective value	infeasible	22,807.10	infeasible

Table 5: Description of the LOCKWOOD problem variables. The coordinates of the best known solution have been rounded.

Variable description	Lower bound	Best known	Upper bound
Wing span b (ft)	30.0	44.19	45.0
Root cord C_r (ft)	6.0	6.75	12.0
Taper ratio λ	0.28	0.28	0.50
Tube external diameter d_r (ft)	1.6	4.03	5.0
Tube thickness t_r (ft)	0.3	0.3	0.79
Angle of attack at root α_r (deg)	-1.0	3.0	3.0
Angle of attack at tip and at rest α_{0t} (deg)	-1.0	0.72	3.0
Objective value	failure	-16.6101	-8.0157

Table 6: Description of the MDO problem variables. The coordinates and the value of the best known solution have been rounded.

4.2.3 Description of the MDO problem

The third and last application is a *multidisciplinary design optimization* (MDO) problem taken from [60]. It simulates the shape and structural design of a conceptual wing design built around a tube. For this, two *disciplines* are used, each of which focuses on a particular aspect of the wing, namely the aerodynamic loading and the structural displacements. Since a modification in one discipline impacts the other discipline, a coupling appears.

The current approach consists to solve this coupling as a separated and stand-alone analysis, called the *multidisciplinary design analysis* (MDA), based on the fixed-point method, similarly to the HS67 problem studied in Section 4.1.2. The outputs of the MDA are the lift L, the drag D, the weight of the fuel W_f, the structural weight of the wing W_s, and the maximum shear and tensile stresses: σ_{max}, τ_{max}.

The problem has 7 optimization variables corresponding to the geometry of the wing. They are described in Table 6, along with bound values and the best known design. This table also gives some values of the objective function to minimize, which is defined by $f(x) = -\frac{L}{D} \log \left(\frac{W_0+W_f+W_s}{W_0+W_s} \right)$ as an approximation of the range of the aircraft. The constant $W_0 = 6,000$ lb corresponds to the airframe weight and payload.

Finally, we consider the three following constraints: $L \geq W_0 + W_f + W_s$ (lift superior to the total weight of the plane), $\sigma_{max} \leq 73,200$ psi, and $\tau_{max} \leq 47,900$ psi (stress constraints).

4.2.4 Results

One evaluation takes about 4 seconds for STYRENE, and 2 seconds for both LOCKWOOD and MDO, which makes these problems convenient candidates for benchmarks. Thus, for these tests, a budget of 1,000 evaluations is considered and convergence to the minimal mesh size is not required.

Table 7: Results for the STYRENE problem ($n = 8$, $m = 11$). The "–" marks are used when the initial LH search could not find a feasible solution.

#	start	MADS alone obj.	MADS alone impr.	Quadratic obj.	Quadratic impr.	TGP obj.	TGP impr.
S01	-22,647,400	-24,175,200	6.32	-24,849,500	8.86	**-33,339,800**	**32.07**
S02	-19,251,700	-32,167,100	40.15	-32,124,300	40.07	**-32,339,400**	**40.47**
S03	-7,778,260	-32,891,000	76.35	**-32,904,300**	**76.36**	-32,891,900	76.35
S04	-14,257,400	**-32,488,500**	**56.12**	-32,444,000	56.06	-31,951,900	55.38
S05	-15,464,300	-30,218,200	48.82	-30,332,800	49.02	**-32,516,400**	**52.44**
S06	–	-32,678,100	–	-32,674,400	–	**-32,939,900**	–
S07	–	-24,566,800	–	-25,885,000	–	**-32,230,900**	–
S08	–	-27,015,800	–	-30,345,000	–	**-31,149,900**	–
S09	–	-23,897,200	–	-23,362,300	–	**-32,276,300**	–
S10	–	-30,786,700	–	-30,592,000	–	**-31,012,700**	–
avg	-15,879,812	-29,088,460	45.55	-29,551,360	46.07	**-32,264,910**	**51.34**

In the present context where n and m are relatively large, the computational cost of TGP can become prohibitive compared to the blackbox cost. To give an idea, one entire optimization performed by MADS–TGP takes approximately 3 hours. This computational cost is mainly due to the use of 500 prediction points along with the computation of the expected reduction in variance. One optimization with MADS alone or with quadratic models takes around 30 minutes to complete.

Tables 7, 8, and 9 present the results on the STYRENE, LOCKWOOD, and MDO problems, respectively, for MADS alone, and for MADS with quadratic and TGP surrogate models. A total of 30 instances have been considered (S01 to S10, L01 to L10, and M01 to M10), with 30 LH points for STYRENE and 20 LH points for LOCKWOOD and MDO as starting guesses. Since the limit of 1,000 evaluations was always reached before convergence, we display the "impr." column instead of "eval." Quadratic models do not offer much improvement over MADS for the STYRENE problem ($\simeq$0.5% on average), probably due to the nonsmooth nature of the functions. TGP, however, shows consistent improvement. For LOCKWOOD, quadratic models fared better. However TGP gave better solutions for six instances out of ten. For MDO, using quadratic models or TGP is always preferable, and MADS–TGP was better for nine instances out of 10.

Figures 9, 10, and 11 show the performance and data profiles for the 30 STYRENE, LOCKWOOD, and MDO instances. There again, the TGP variant dominates the two other methods. The profiles illustrate also that using quadratic models for STYRENE does not improve the MADS alone version, and again, it is likely due to the nonsmoothness of the problem.

Finally, Table 10 and Figure 12 illustrate the impact of the maximum number of search points introduced in Section 3.4. In `NOMAD`, this corresponds to the `MODEL_SEARCH_MAX_TRIAL_PTS` option. These results are given here for the MDO problem only, but unreported empirical tests gave the same conclusion. We used the exact same 10 instances M01 to M10 as in Table 9 and Figure 11. Three values are compared: 2, 5, and 10 points. The results show that 10 points is generally better, and this is the reason why it has been taken as the default value for all the other tests and in `NOMAD`.

Table 8: Results for the LOCKWOOD problem ($n = 6$, $m = 2$).

#	start	MADS alone		Quadratic		TGP	
		obj.	impr.	obj.	impr.	obj.	impr.
L01	60,000.00	27,211.40	54.65	34,057.00	43.24	**24,700.20**	**58.83**
L02	39,593.60	24,881.80	37.16	25,961.60	34.43	**22,807.10**	**42.40**
L03	37,233.10	27,195.20	26.96	**22,989.70**	**38.25**	28,557.40	23.30
L04	44,566.50	25,674.30	42.39	26,259.40	41.08	**25,037.60**	**43.82**
L05	45,309.60	27,382.30	39.57	27,045.60	40.31	**22,973.50**	**49.30**
L06	60,000.00	28,467.20	52.55	**23,336.00**	**61.11**	25,161.50	58.06
L07	60,000.00	29,262.50	51.23	**25,784.00**	**57.03**	26,456.60	55.91
L08	47,228.20	23,834.00	49.53	23,293.20	50.68	**22,880.40**	**51.55**
L09	46,778.20	25,313.20	45.89	24,224.30	48.21	**23,276.00**	**50.24**
L10	51,490.20	26,642.70	48.26	**23,287.00**	**54.77**	25,618.90	50.25
avg	49,219.94	26,586.46	44.82	25,623.78	46.91	**24,746.92**	**48.37**

Table 9: Results for the MDO problem ($n = 7$, $m = 3$). The "–" marks are used when the initial LH search could not find a feasible solution.

#	start	MADS alone		Quadratic		TGP	
		obj.	impr.	obj.	impr.	obj.	impr.
M01	–	-16.1283	–	-16.1069	–	**-16.2173**	–
M02	-10.9119	-15.8245	45.02	-16.0678	47.25	**-16.1118**	**47.65**
M03	-12.2374	-16.3364	33.50	-16.2661	32.92	**-16.5283**	**35.06**
M04	–	-15.9130	–	**-16.0973**	–	-15.9017	–
M05	–	-16.3483	–	-16.3389	–	**-16.5773**	–
M06	-10.6734	-16.1393	51.21	-16.0301	50.19	**-16.2588**	**52.33**
M07	-12.3317	-15.9438	29.29	-16.5325	34.07	**-16.5923**	**34.55**
M08	–	-16.1559	–	-16.2779	–	**-16.5912**	–
M09	–	-16.1344	–	-16.2095	–	**-16.6057**	–
M10	-13.8869	-15.9013	14.51	-16.3043	17.41	**-16.3361**	**17.64**
avg	-12.0082	-16.0825	34.70	-16.2231	36.37	**-16.3721**	**37.45**

Table 10: Impact of the number of search points for the MDO problem ($n = 7$, $m = 3$). The "–" marks are used when the initial LH search could not find a feasible solution.

#	start	TGP 2		TGP 5		TGP 10 (default)	
		obj.	impr.	obj.	impr.	obj.	impr.
M01	–	-15.9026	–	-15.9707	–	**-16.2173**	–
M02	-10.9119	-16.5429	51.60	**-16.5684**	**51.84**	-16.1118	47.65
M03	-12.2374	**-16.6091**	**35.72**	-16.3479	33.59	-16.5283	35.06
M04	–	**-16.4508**	–	-16.2709	–	-15.9017	–
M05	–	-16.4460	–	-16.2199	–	**-16.5773**	–
M06	-10.6734	-16.5986	55.51	**-16.6018**	**55.54**	-16.2588	52.33
M07	-12.3317	-16.4298	33.23	-16.4718	33.57	**-16.5923**	**34.55**
M08	–	**-16.6089**	–	-16.5851	–	-16.5912	–
M09	–	-16.5827	–	-16.6019	–	**-16.6057**	–
M10	-13.8869	-16.3049	17.41	**-16.3849**	**17.99**	-16.3361	17.64
avg	-12.0082	**-16.4476**	**38.70**	-16.4023	38.51	-16.3721	37.45

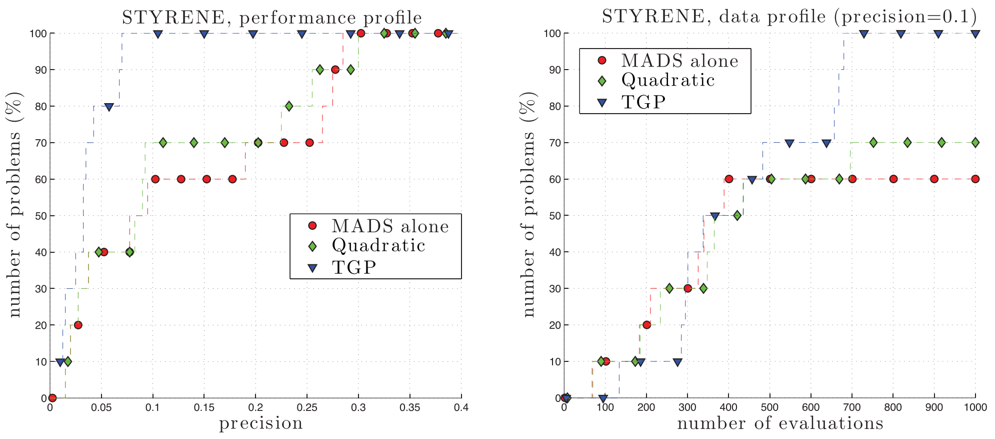

Figure 9: Performance (left) and data (right) profiles for the STYRENE problem ($n = 8$, $m = 11$).

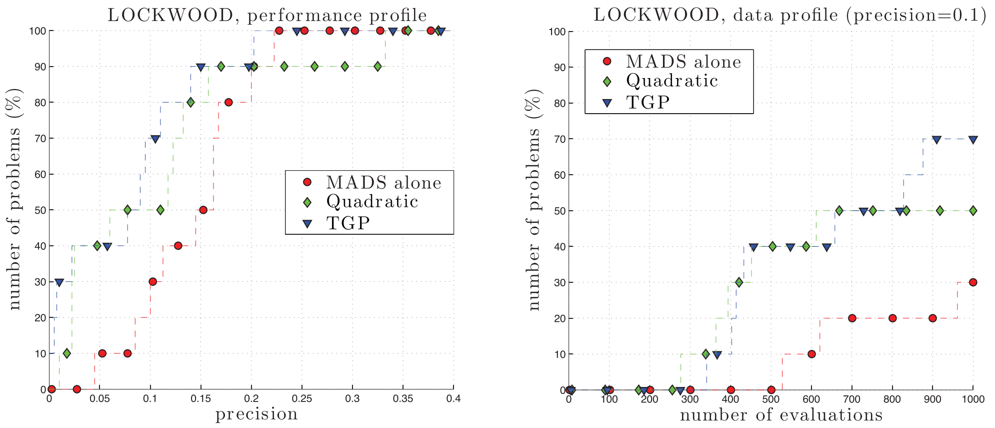

Figure 10: Performance (left) and data (right) profiles for the LOCKWOOD problem ($n = 6$, $m = 2$).

$\boxed{5}$ Discussion

We have presented the use of TGP surrogates within the MADS framework in order to improve the MADS algorithm for blackbox optimization. The efficiency of the new method has been demonstrated for five test cases: one synthetic but with many local optima, another synthetic but close to real-world, and three real-data applications.

Future projects include ways of address the issue of the TGP computing cost in the context of fast blackbox simulations while keeping the best interpolation quality as possible. Future work also includes the use of classification tools for modeling the set of non-quantifiable and/or unrelaxable constraints $\mathcal{X}$ [30, 44], the use of a derivative-based solver for surrogate optimization, the study of a cooperative use of TGP surrogates and quadratic models, and finally the application to other real-life problems.

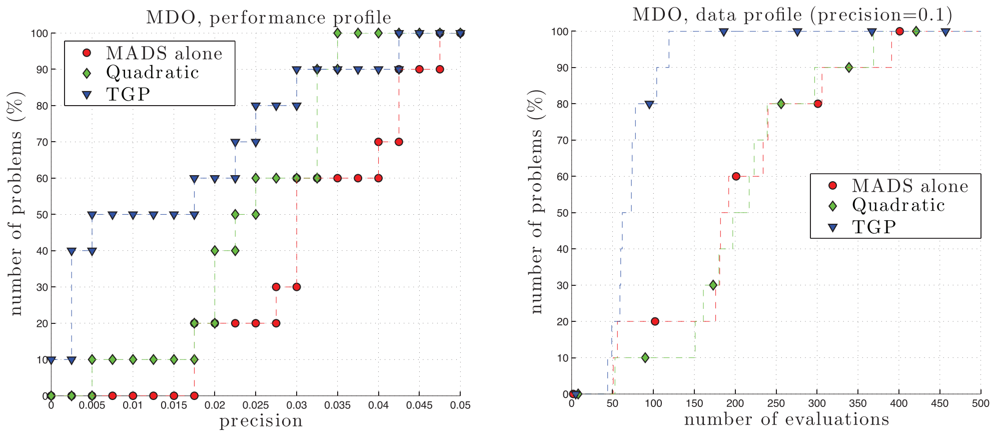

Figure 11: Performance (left) and data (right) profiles for the MDO problem ($n = 7$, $m = 3$). In the data profile, the plots are stopped at 500 evaluations.

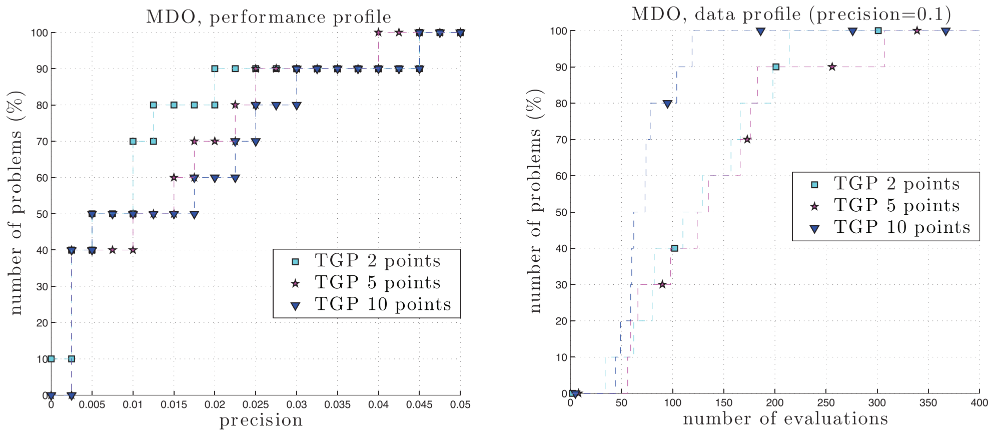

Figure 12: Impact of the number of search points for the MDO problem ($n = 7$, $m = 3$). In the data profile, the plots are stopped at 400 evaluations.

Acknowledgments

Both authors would like to thank the American Institute of Mathematics for hosting them in a series of small group meetings. We also would like to thank Charles Audet, Dominique Orban, Bastien Talgorn, and Mark Abramson for their useful comments, Shawn Matott, Genetha Gray, and Stefan Wild for making the LOCKWOOD problem available, Andrew Booker for indicating references [10, 51], and Phillipe R. Sampaio and three anonymous referees for the constructive remarks that helped to improve this paper.

References

[1] P. Abrahamsen, A Review of Gaussian random fields and correlation functions, Norwegian Computing Center, Box 114 Blindern, N-0314 Oslo, Norway, No. 917, 1997.

[2] M.A. Abramson, C. Audet, G. Couture, J.E. Dennis, Jr., S. Le Digabel and C. Tribes, The NOMAD project, Software available at `http://www.gerad.ca/nomad`.

[3] M.A. Abramson, C. Audet, J.E. Dennis, Jr.f and S. Le Digabel, A deterministic MADS instance with orthogonal directions, *SIAM Journal on Optimization* 20 (2009) 948–966.

[4] C. Audet, V. Béchard and S. Le Digabel, Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search, *Journal of Global Optimization* 41 (2008) 299–318.

[5] C. Audet, A.L. Custódio and J.E. Dennis, Jr., Erratum: Mesh Adaptive Direct Search Algorithms for Constrained Optimization, *SIAM Journal on Optimization* 4 (2008) 1501–1503.

[6] C. Audet and J.E. Dennis, Jr., Mesh adaptive direct search algorithms for constrained optimization, *SIAM Journal on Optimization* 17 (2006) 188–217.

[7] C. Audet and J.E. Dennis, Jr., A progressive barrier for derivative-free nonlinear programming, *SIAM Journal on Optimization* 20 (2009) 445–472.

[8] C. Audet, J.E. Dennis, Jr. and S. Le Digabel, Parallel space decomposition of the mesh adaptive direct search algorithm, *SIAM Journal on Optimization* 3 (2008) 1150–1170.

[9] C. Audet, J.E. Dennis, Jr. and S. Le Digabel, Globalization strategies for mesh adaptive direct search, *Computational Optimization and Applications* 46 (2010) 193–215.

[10] S. Ba and V.R. Joseph, Composite Gaussian process for emulating expensive functions, *Annals of Applied Statistics* 6 (2012) 1838–1860.

[11] A.J. Booker, Well-conditioned Kriging models for optimization of computer simulations, Technical Report M&CT-TECH-00-002, Boeing Computer Services, Research and Technology, M/S 7L–68, Seattle, Washington 98124, 2000.

[12] A.J. Booker, J.E. Dennis, Jr., P.D. Frank and D.B. Serafini, V. Torczon and M.W. Trosset, A rigorous framework for optimization of expensive functions by surrogates, *Structural and Multidisciplinary Optimization* 17 (1999) 1–13.

[13] R. Brekelmans, L. Driessen, H. Hamers and D. den Hertog, Constrained optimization involving expensive function evaluations: A sequential approach, *European Journal of Operational Research* 160 (2005) 121–138.

[14] A.D. Bull, Convergence rates of efficient global optimization algorithms, *Journal of Machine Learning Research* 12 (2011) 2879–2904.

[15] K.H. Chang, Stochastic Nelder-Mead simplex method - A new globally convergent direct search method for simulation optimization, *European Journal of Operational Research* 220 (2012) 684–694.

[16] Y.G. Chen, S. Lu, X.L. Liu and Y.W. Chen, Derivative-free hybrid optimization method for top design of satellite system, *Advanced Materials Research* 308–310 (2011) 2413–2417.

[17] H.A. Chipman, E.I. George and R.E. McCulloch, Bayesian CART model search (with discussion), *Journal of the American Statistical Association* 93 (1998) 935–960.

[18] H.A Chipman, E.I. George and R.E. McCulloch, Bayesian treed models, *Machine Learning* 48 (2002) 299–320.

[19] F.H. Clarke, *Optimization and Nonsmooth Analysis*, John Wiley & Sons, New York, 1983.

[20] A.R. Conn and S. Le Digabel, Use of quadratic models with mesh-adaptive direct search for constrained black box optimization, *Optimization Methods and Software* 28 (2013) 139–158.

[21] A.R. Conn, K. Scheinberg and L.N. Vicente, *Introduction to Derivative-Free Optimization*, MOS/SIAM Series on Optimization, SIAM, Philadelphia, 2009.

[22] J. Craig, Bluebird Developer Manual, 2002, Available at `http://www.civil.uwaterloo.ca/jrcraig/pdf/bluebird_developers_manual.pdf`.

[23] A.L. Custódio, H. Rocha and L.N. Vicente, Incorporating minimum Frobenius norm models in direct search, *Computational Optimization and Applications* 46 (2010) 265–278.

[24] A.L. Custódio and L.N. Vicente, Using sampling and simplex derivatives in pattern search methods, *SIAM Journal on Optimization* 18 (2007) 537–555.

[25] R. Fletcher and S. Leyffer, Nonlinear programming without a penalty function, *Mathematical Programming* Series A, 91 (2002) 239–269.

[26] N.I.M. Gould, D. Orban and Ph.L. Toint, A constrained and unconstrained testing environment with safe threads, Technical Report G-2013-27, Les cahiers du GERAD, 2013. To appear in *Computational Optimization and Applications.*

[27] R.B. Gramacy An R Package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models, *Journal of Statistical Software* 19 (2007) 1–46.

[28] R.B. Gramacy and H.K.H. Lee, Bayesian treed Gaussian process models with an application to computer modeling, *Journal of the American Statistical Association* 103 (2008) 1119–1130.

[29] R.B. Gramacy and H.K.H. Lee, *Adaptive design and analysis of supercomputer experiment*, Technometrics 51 (2009) 130–145.

[30] R.B. Gramacy and H.K.H. Lee, Optimization under unknown constraints, in Proceedings of the ninth Valencia International Meetings on Bayesian Statistics, J. Bernardo, S. Bayarri, J.O. Berger, A.P. Dawid, D. Heckerman, A.F.M. Smith and M. West (eds), Oxford University Press, 2011, pp. 229–256.

[31] R.B. Gramacy and H.K.H. Lee, Cases for the nugget in modeling computer experiments, *Statistics and Computing* 22 (2012) 713–722.

[32] R.B. Gramacy and N.G. Polson, Particle learning of Gaussian process models for sequential design and optimization, *Journal of Computational and Graphical Statistics* 20 (2011) 102–118.

[33] R.B. Gramacy and M.A. Taddy, Categorical Inputs, sensitivity analysis, optimization and importance tempering with `tgp` Version 2, an `R` packge for treed Gaussian process models, *Journal of Statistical Software* 33 (2010) 1–48.

[34] G.A. Gray and T.G. Kolda, Algorithm 856: APPSPACK 4.0: Asynchronous parallel pattern search for derivative-free optimization, *ACM Transactions on Mathematical Software*, 32 (2006) 485–507.

[35] A.-R. Hedar, Global optimization test problems, `http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm`.

[36] A.-R. Hedar and M. Fukushima, Tabu search directed by direct search methods for nonlinear global optimization, *European Journal of Operational Research* 170 (2006) 329–349.

[37] A.F. Hernandez and M.A. Grover, Error estimation properties of Gaussian process models in stochastic simulations, *European Journal of Operational Research* 228 (2013) 131–140.

[38] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, vol. 187, Lecture Notes in Economics and Mathematical Systems, 1981, Springer Verlag, Berlin, 1981.

[39] L.M. Hvattum and F. Glover, Finding local optima of high-dimensional functions using direct search methods, *European Journal of Operational Research* 195 (2009) 31–45.

[40] D.R Jones, M. Schonlau and W.J. Welch, Efficient global optimization of expensive black box functions, *Journal of Global Optimization* 13 (1998) 455–492.

[41] D.G. Krige, A statistical approach to some mine valuations and allied problems at the Witwatersrand, University of Witwatersrand, 1951.

[42] S. Le Digabel, Nonlinear optimization with the MADS algorithm, *ACM Transactions on Mathematical Software* 37 (2011) 44:1–44:15.

[43] S. Le Digabel, M.A. Abramson, C. Audet and J.E. Dennis, Jr., Parallel Versions of the MADS Algorithm for Black-Box Optimization, in Optimization Days, GERAD, Montreal, 2010.

[44] H.K.H Lee, R.B. Gramacy, C. Linkletter and G.A. Gray, Optimization Subject to Hidden Constraints via Statistical Emulation, *Pacific Journal of Optimization* 8 (2011) 467–478.

[45] A.L. Marsden, J.A. Feinstein and C.A. Taylor, A computational framework for derivative-free optimization of cardiovascular geometries, *Computer Methods in Applied Mechanics and Engineering* 197 (2008) 1890–1905.

[46] A.L. Marsden, M. Wang, J.E. Dennis, Jr. and P. Moin, Trailing-edge noise reduction using derivative-free optimization and large-eddy simulation, *Journal of Fluid Mechanics* 572 (2007) 13–36.

[47] L.S. Matott, K. Leung and J. Sim, Application of MATLAB and Python optimizers to two case studies involving groundwater flow and contaminant transport modeling, *Computers & Geosciences* 37 (2011) 1894–1899.

[48] M.D. McKay, R.J. Beckman and W.J. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics* 21 (1979) 239–245.

[49] J.J. Moré and S.M. Wild, Benchmarking derivative-free optimization algorithms, *SIAM Journal on Optimization* 20 (2009) 172–191.

[50] D. Orban, Templating and automatic code generation for performance with Python, Les cahiers du GERAD, G-2011-30, `https://www.gerad.ca/~orban/papers.html`, 2011.

[51] P. Ranjan, R. Haynes and R. Karsten, A computationally stable approach to Gaussian process interpolation of deterministic computer simulation data, *Technometrics* 53 (2011) 366–378.

[52] C.E. Rasmussen and C.K.I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[53] R.G. Regis and C.A. Shoemaker, Parallel radial basis function methods for the global optimization of expensive functions, *European Journal of Operational Research* 182 (2007) 514–535.

[54] T.J. Santner, B.J. Williams and W.I. Notz, *The Design and Analysis of Computer Experiments*, Springer, New York, NY, 2003.

[55] S. Seo, M. Wallat, T. Graepel and K. Obermayer, Gaussian process regression: active data selection and test point rejection, in *Proceeings of the International Joint Conference on Neural Networks*, Vol. III, IEEE, 2000, pp. 241–246.

[56] M.A. Taddy, R.B. Gramacy and N. Polson, Dynamic trees for learning and design, *Journal of the American Statistical Association* 106 (2011) 109–123.

[57] M.A. Taddy, H.K.H. Lee, G.A. Gray and J.D. Griffin, Bayesian guided pattern search for robust local optimization, *Technometrics* 51 (2009) 389–401.

[58] Development Core Team, *A Language and Environment for Statistical Computing*, Foundation for Statistical Computing, Vienna, 2004.

[59] V. Torczon, On the convergence of pattern search algorithms, *SIAM Journal on Optimization* 7 (1997) 1–25.

[60] C. Tribes, J.-F. Dubé and J.-Y. Trépanier, Decomposition of multidisciplinary optimization problems: formulations and application to a simplified wing design, *Engineering Optimization*, 37 (2005) 775–796.

[61] A.I.F. Vaz and L.N. Vicente, A particle swarm pattern search method for bound constrained global optimization, *Journal of Global Optimization* 39 (2007) 197–219.

Manuscript received 6 September 2013
accepted for publication 7 August 2014

Robert B. Gramacy
Kemper Family Foundation Scholar, 2011–2012
The University of Chicago Booth School of Business
5807 South Woodlawn Avenue, Chicago, Illinois 60637, USA
E-mail address: `rbgramacy@chicagobooth.edu`

Sébastien Le Digabel
GERAD and Département de mathématiques et génie industriel
École Polytechnique de Montréal
C.P. 6079, Succursale Centre-ville
Montréal, Québec H3C 3A7, Canada
E-mail address: `sebastien.le.digabel@gerad.ca`